Towards Universal Unfolding using Denoising Diffusion Probabilistic Models - ML4Jets Paris

Camila Pazos, Shuchin Aeron, Pierre-Hugues Beauchemin, Vincent Croft, Zhengyan Huan, Martin Klassen, Taritree Wongjirad

Unfolding

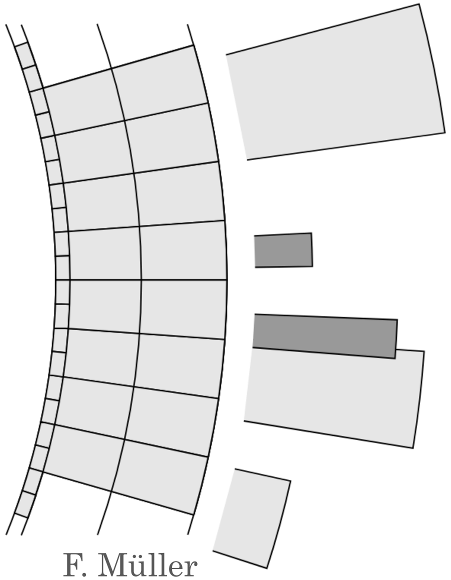
- Want to obtain truth-level kinematics distribution $f_{\mathrm{true}}(\mathbf{x})$
- We measure

$$f_{\mathrm{det}}(\mathbf{y}) = \int d\mathbf{x}\, P(\mathbf{y}|\mathbf{x})\, f_{\mathrm{true}}(\mathbf{x})$$

where $P(\mathbf{y}|\mathbf{x})$ incorporates the detector effects

- Unfolding requires the inverse process

$$P(\mathbf{x}|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{x}) f_{\mathrm{true}}(\mathbf{x})}{f_{\mathrm{det}}(\mathbf{y})}$$



$\mathbf{y}$

Unfolding Challenges

- Unfolded distributions are typically binned
    - ML allows for event-wise unfolding (generative, re-weighting or distribution mapping methods)
- Dependence on the MC prior
- Processes have different detector response

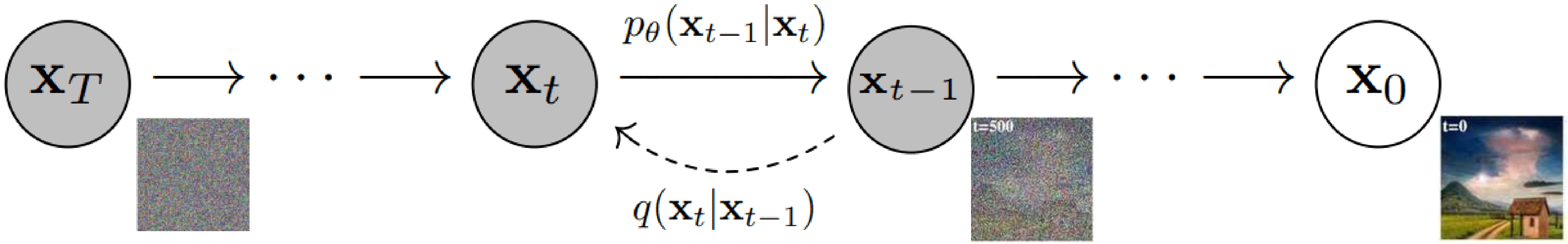
Denoising Diffusion Probabilistic Models

Forward diffusion process:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t\,;\sqrt{1-\beta_t}\,\mathbf{x}_t,\ \beta_t\,\mathbf{I})$$

Reversed denoising process:

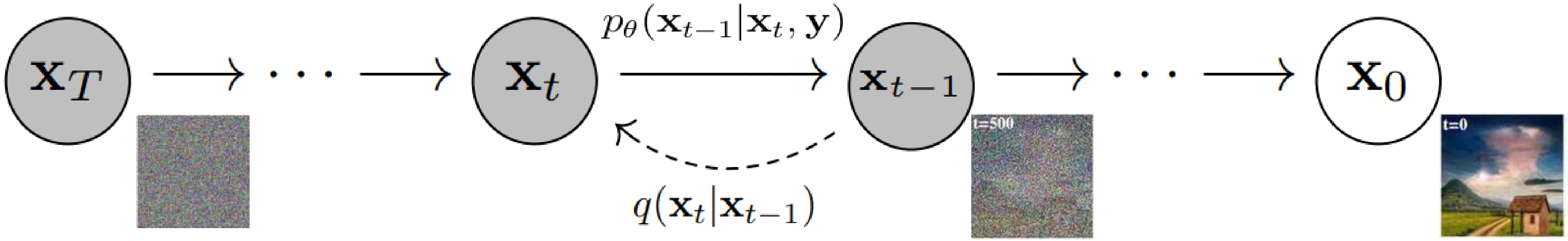
$$p_\theta\left(\mathbf{x}_{0:T}\right) := p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$$



Conditional Denoising Diffusion Probabilistic Models

For unfolding condition on detector measured observables $\mathbf{y}$

$$p_\theta\left(\mathbf{x}_{0:T}|\mathbf{y}\right) := p(\mathbf{x}_T|\mathbf{y}) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}\right)$$



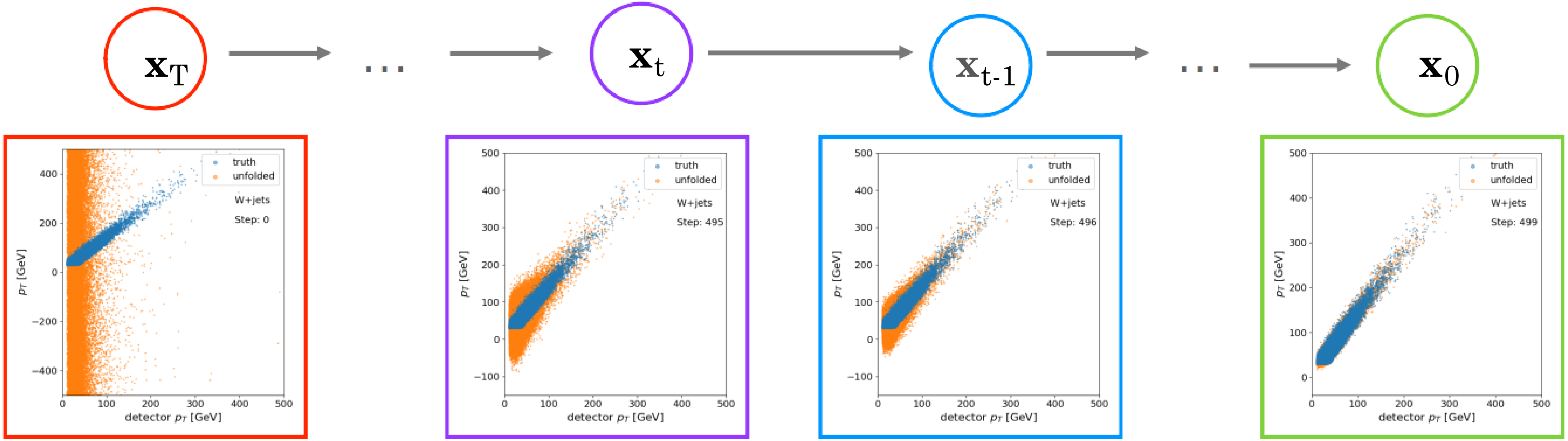
Results

Toy model

Physics cases

Setup

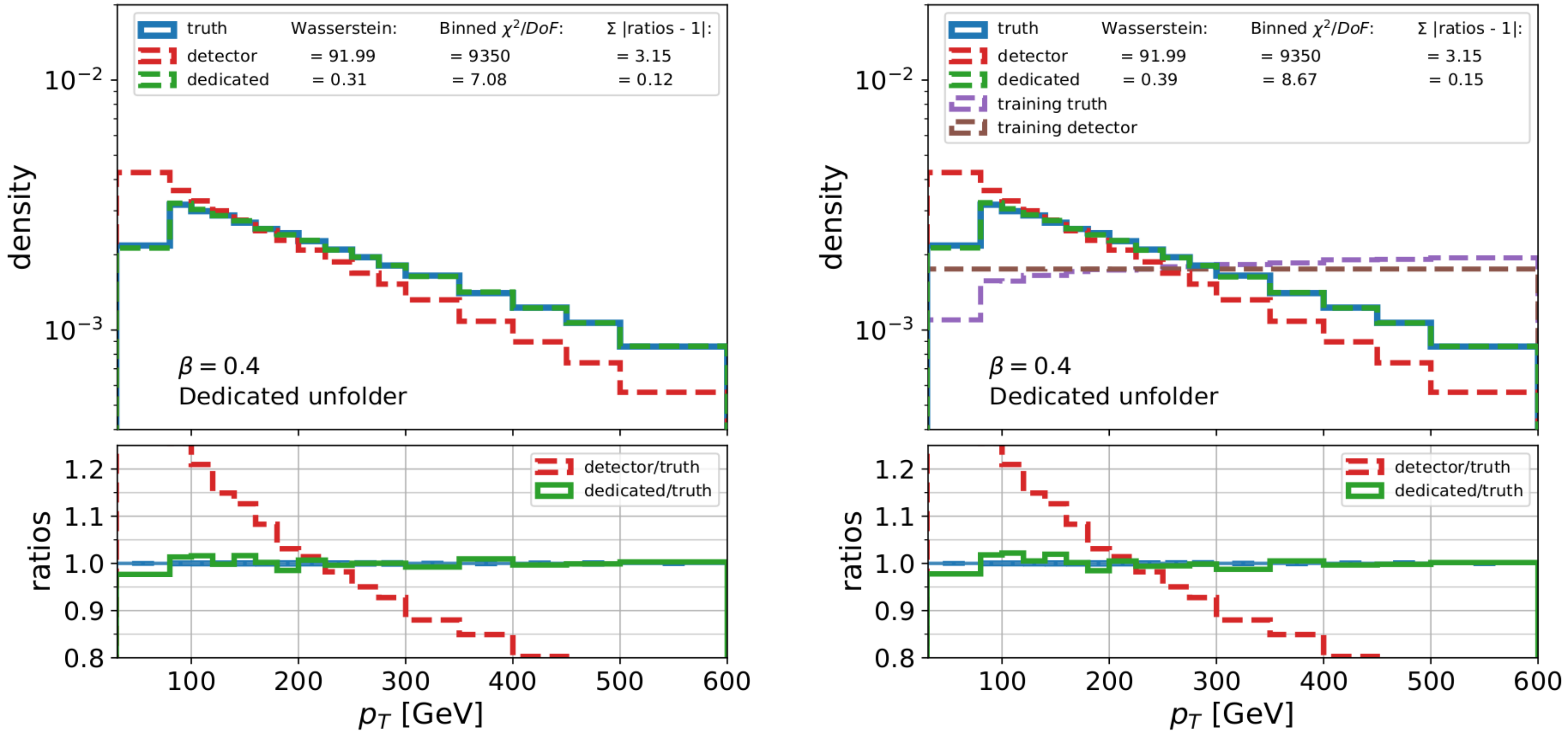
- MLP network
- 7 component jet vector $[p_T,\eta,\phi,E,p_x,p_y,p_z,]$ at truth-level $\mathbf{x}$ and detector-level $\mathbf{y}$
- Training of cDDPM with pairs $\{\mathbf{x},\mathbf{y}\}$ to learn to sample from $P(\mathbf{x}|\mathbf{y})$
- Custom detector simulation using ATLAS response



Toy Model

$$f(x;1/\beta) = (1/\beta)\exp(-x/\beta)$$

Same posteriors:
$P_i(\mathbf{x}\,|\,\mathbf{y}) = P_j(\mathbf{x}\,|\,\mathbf{y})$

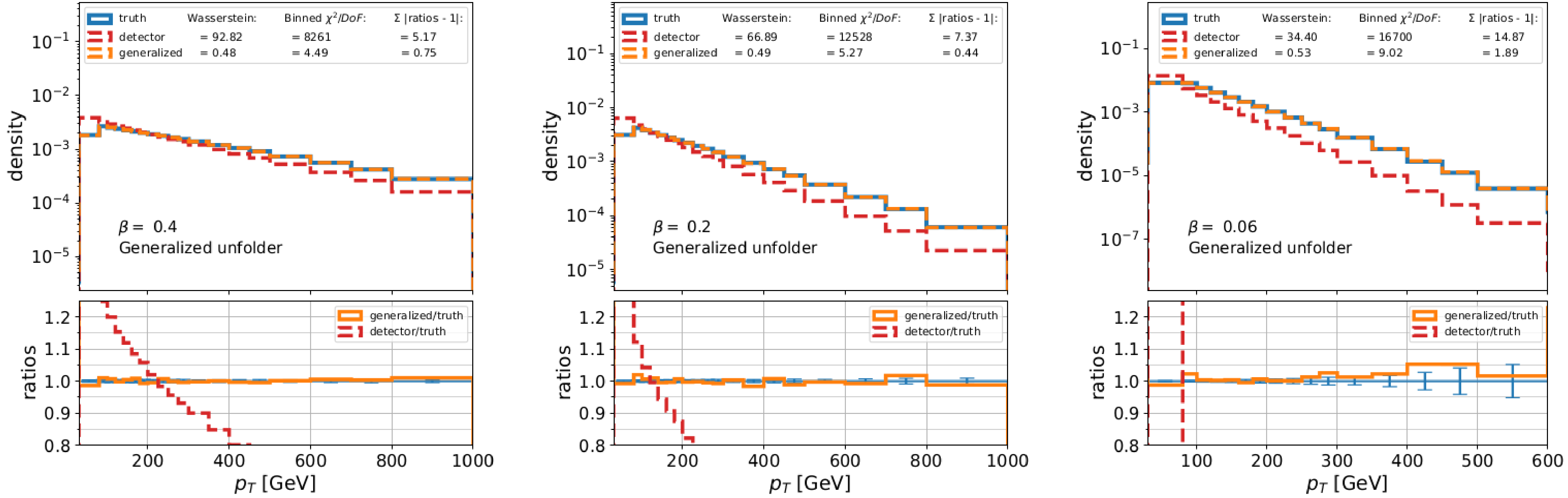


Introduction of Moments

Append the vector by 1st to 6th moment of the p_T distribution:

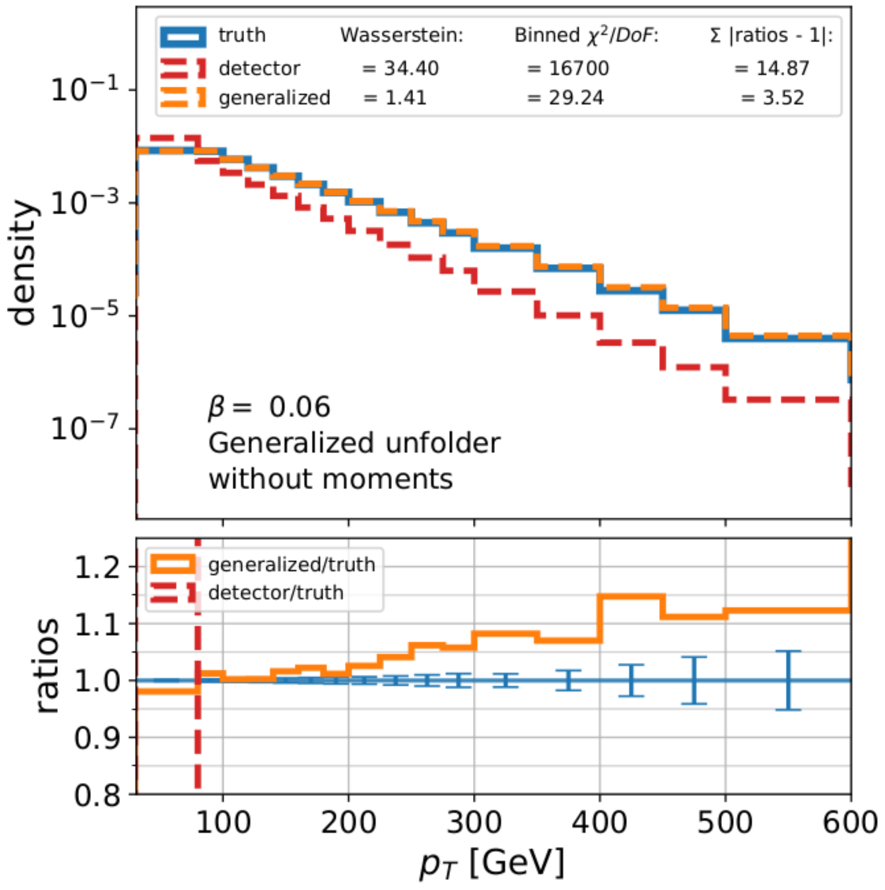
$$\mu = \frac{1}{N}\sum_{i=1}^{N} p_{T,i} \quad \& \quad \mu_k = \frac{1}{N}\sum_{i=1}^{N}(p_{T,i} - \mu)^k$$

$$\frac{P_i(\mathbf{x}|\mathbf{y})}{P_j(\mathbf{x}|\mathbf{y})} = \frac{f^{\,i}_{\text{true}}(\mathbf{x})\, f^{\,j}_{\text{det}}(\mathbf{y})}{f^{\,i}_{\text{det}}(\mathbf{y})\, f^{\,j}_{\text{true}}(\mathbf{x})}$$

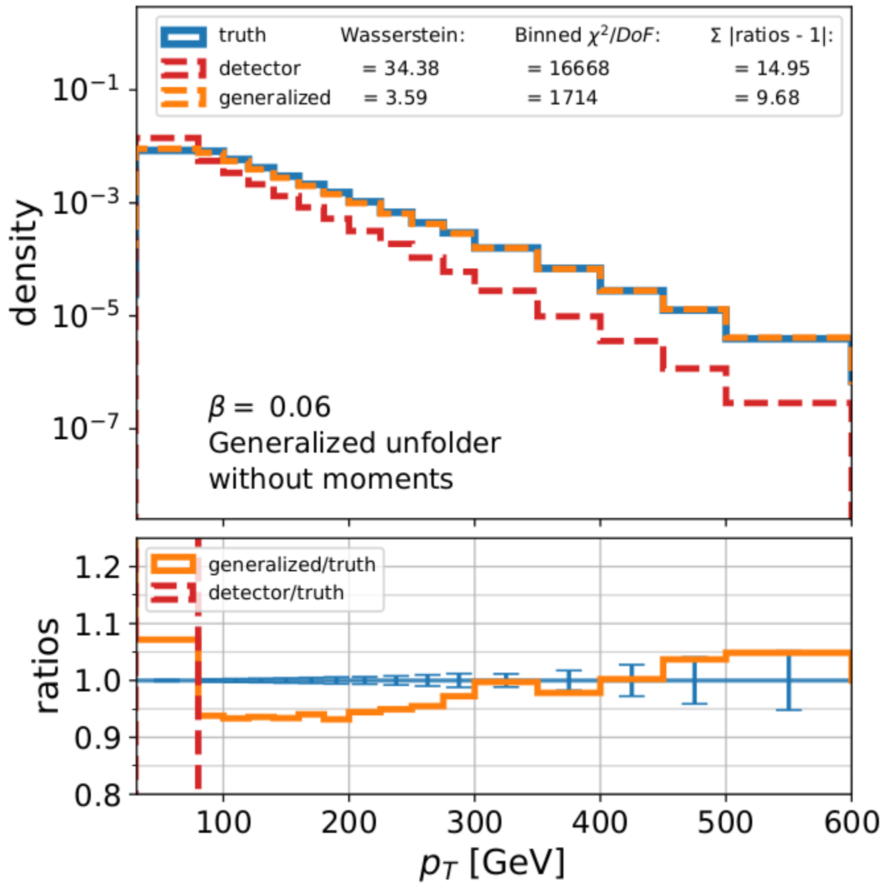


Effect of Moments

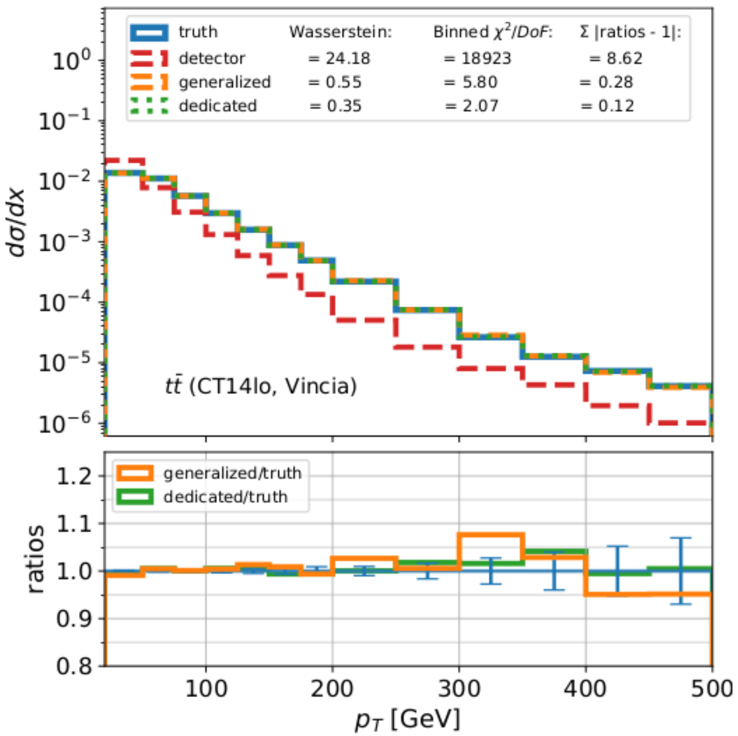
No moments

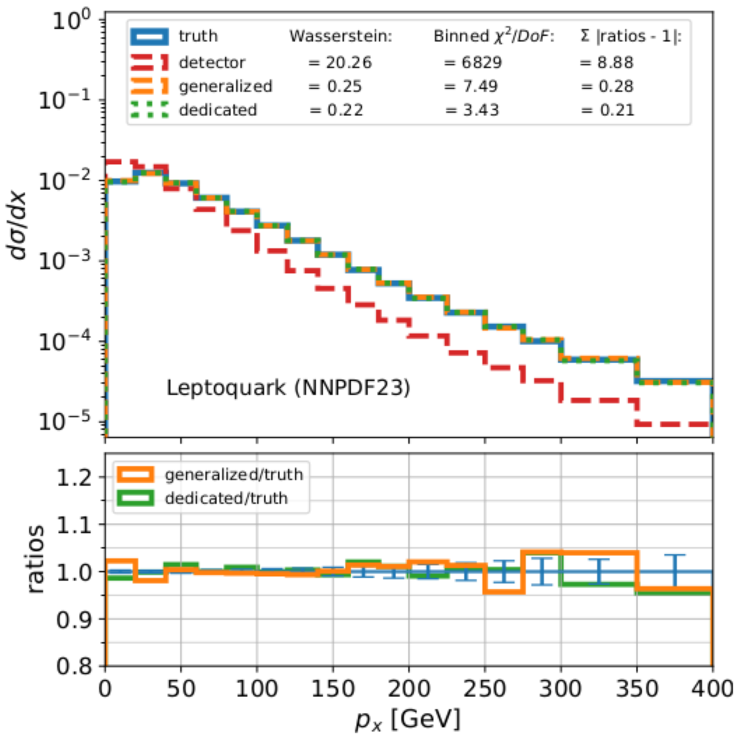


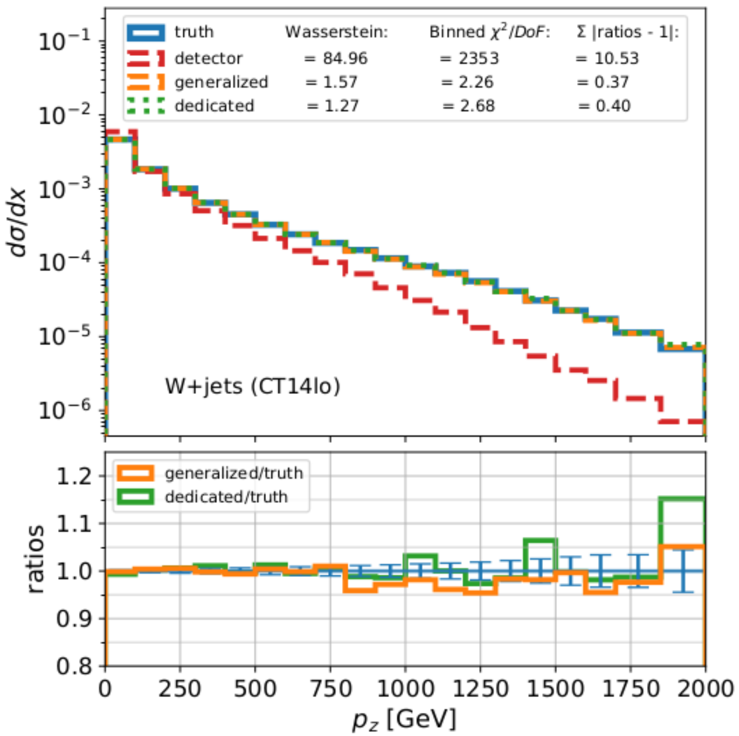
Fake moments



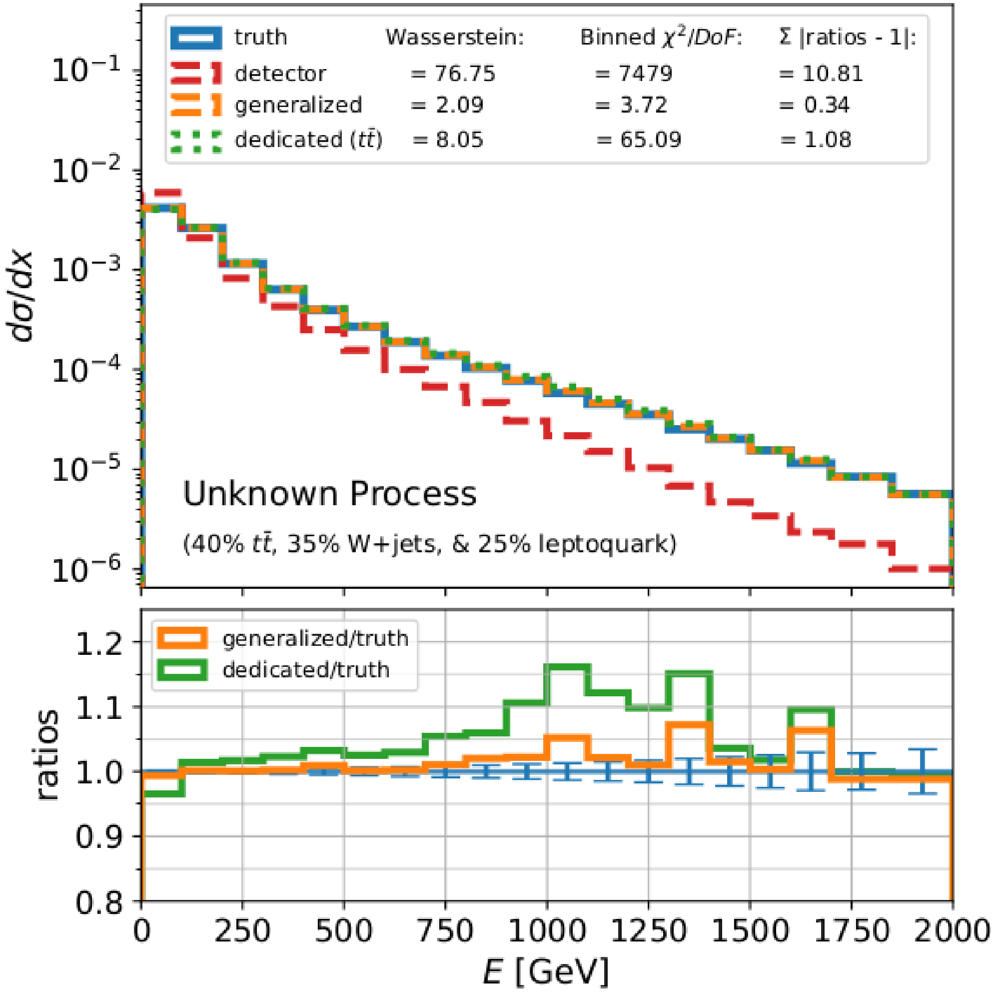
Unfolding Custom Jet Smearing

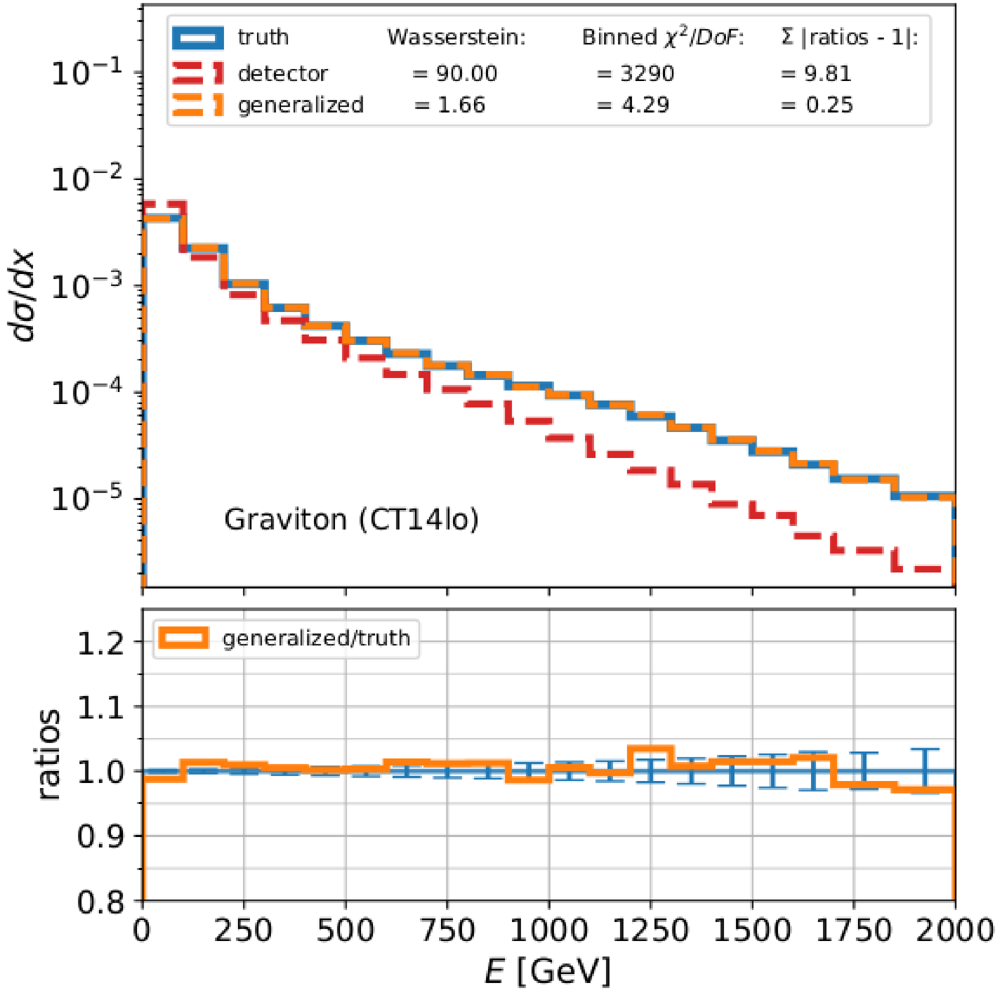




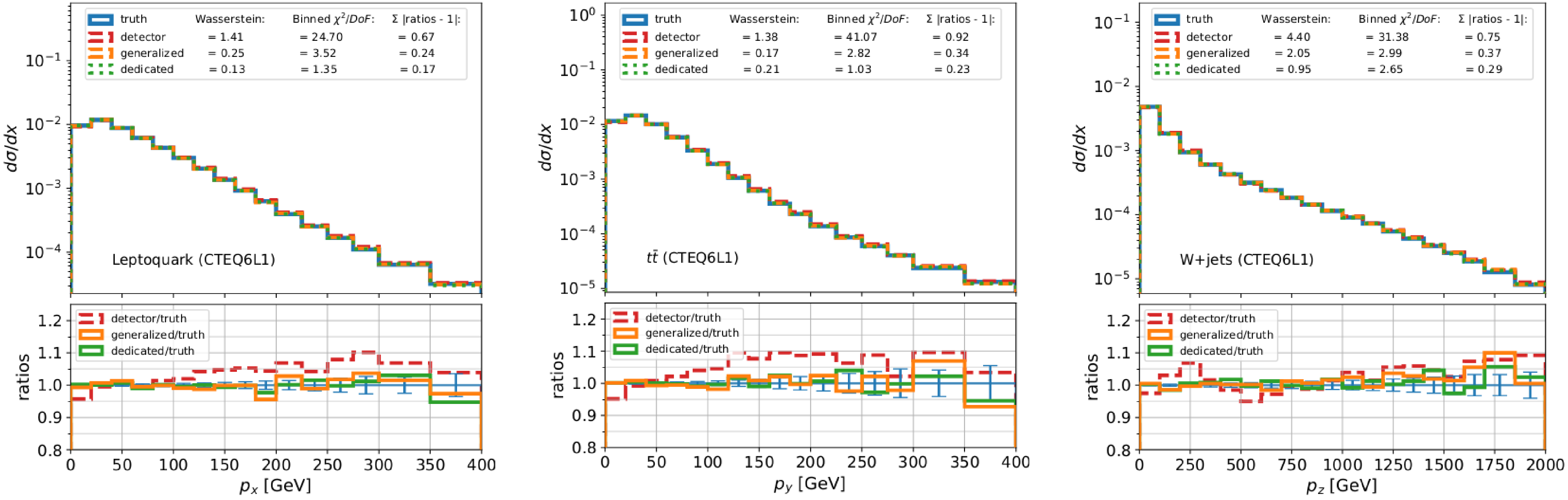


Unfolding Custom Jet Smearing

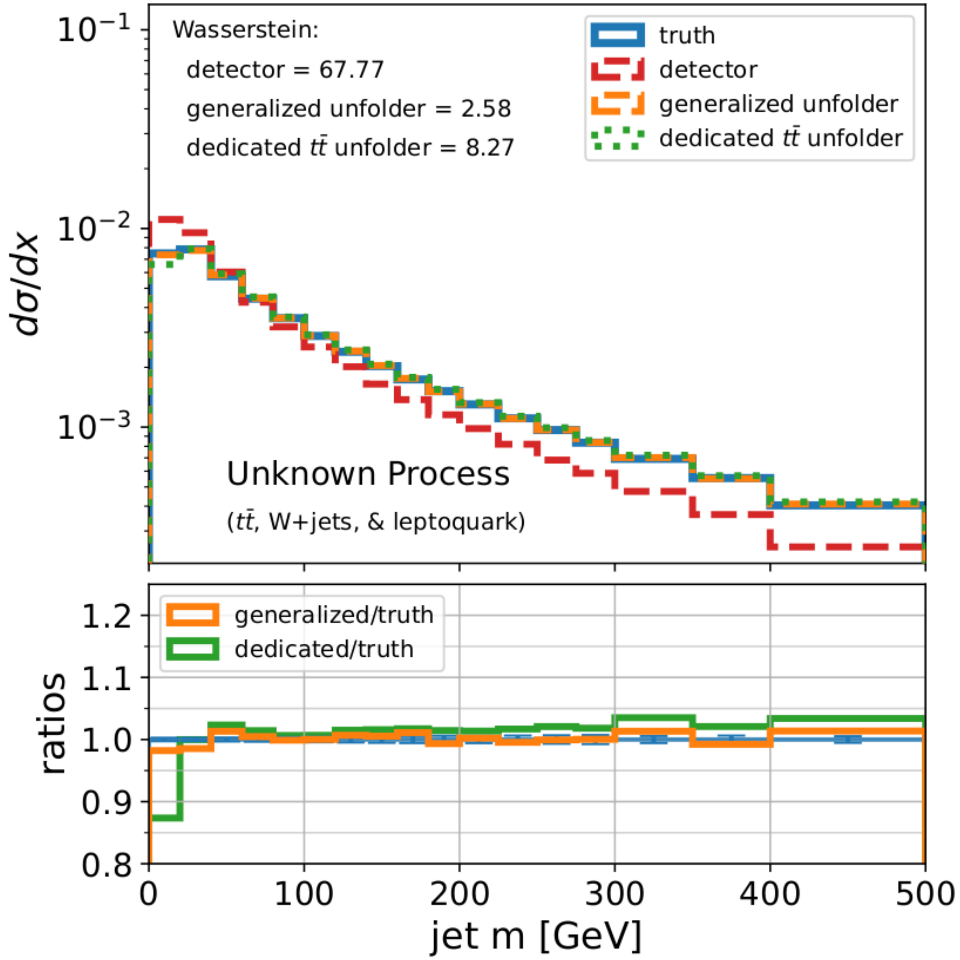


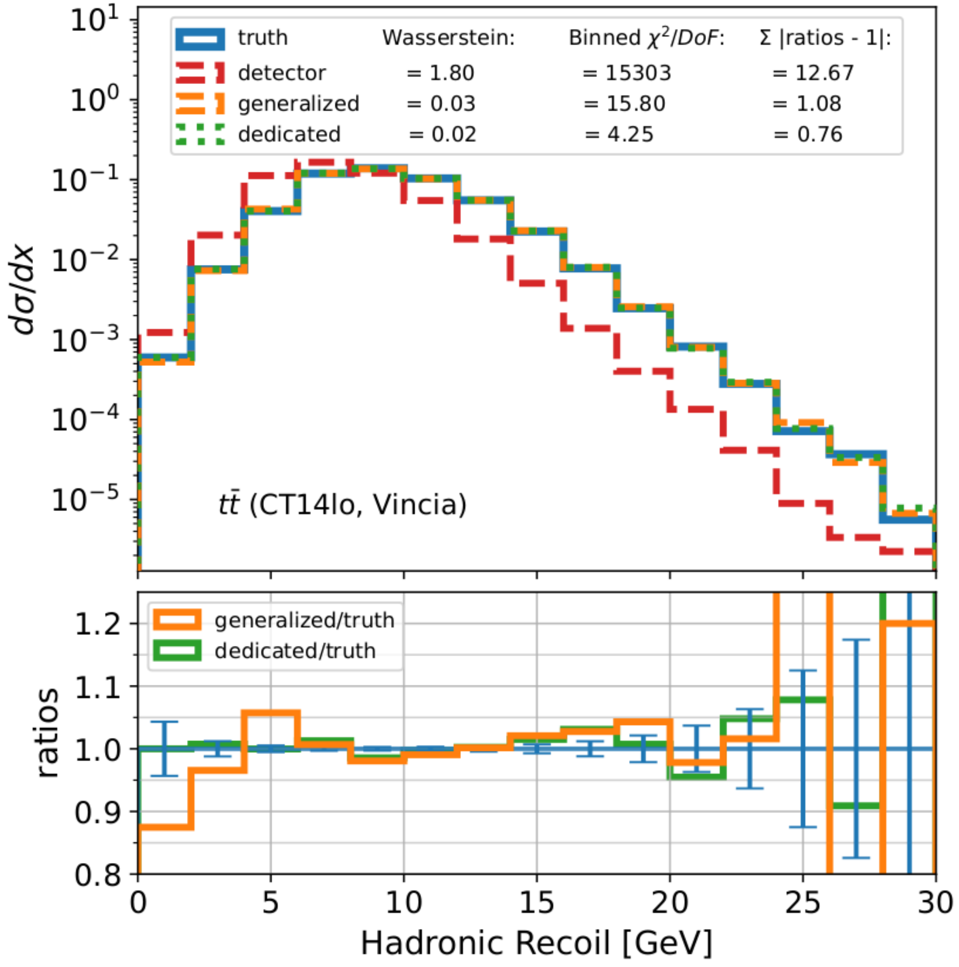


Unfolding Delphes Simulation



Unfolding of Object and Event Observables





Summary

- Successful object-wise unfolding of processes seen and not seen during the training via inclusion of moments possible
- Object correlations conserved
- Possibility of unfolding of unknown processes or without background subtraction
- Dedicated and generalised options available depending on use case

- Future work:
  - Expand this work to other particles than jets
  - Address out of phase space effects
  - Improvements for event correlations required
  - Systematic effects

Datasets

Process	PDF with Parton Shower (Phase Space Bias)	In Training?
$t\bar{t}$	CT14lo	✓
	CT14lo (biased)	✓
	CT14lo with Vincia	
	NNPDF23_lo	✓
	CTEQ6L1	✓
	CTEQ6L1 (biased)	✓
Z+jets	CT14lo	✓
	CT14lo (biased)	✓
	NNPDF23_lo	✓
	CTEQ6L1	
	CTEQ6L1 (biased)	✓
W+jets	CT14lo	
	CT14lo (biased)	✓
	NNPDF23_lo	✓
	CTEQ6L1	✓
Dijets	CT14lo	✓
	CTEQ6L1	✓
	CTEQ6L1 (biased)	✓
Leptoquark	CT14lo	✓
	CT14lo (biased)	✓
	NNPDF23_lo	
	CTEQ6L1	✓

Delphes

Process	PDF (Phase Space Bias)	In Training?
$t\bar{t}$	CTEQ6L1	
	CTEQ6L1 (biased)	✓
Z+jets	CTEQ6L1	✓
	CTEQ6L1 (biased)	✓
W+jets	CTEQ6L1	
	CTEQ6L1 (biased)	✓
Dijets	CTEQ6L1	✓
	CTEQ6L1 (biased)	✓
Leptoquark	CTEQ6L1	

Algorithm

Algorithm 1 Conditional DDPM: Training

Input: dataset $\{\mathbf{x}_0, \mathbf{y}\}$, variance schedule $\beta_1, \ldots \beta_T$

$t \leftarrow \text{Uniform}(\{1, \ldots, T\})$

$\bar{\alpha}_t \leftarrow \prod_{s=1}^{t}(1 - \beta_s)$

$\boldsymbol{\epsilon} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$

Repeat

a) $\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$

b) Calculate loss, $L = ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(t, \mathbf{x}_t, \mathbf{y}\right)||^2$

c) Update θ via $\nabla_\theta L$

Until converged

Algorithm 2 Conditional DDPM: Sampling

Input: detector-level data vector $\mathbf{y}$, variance schedule $\beta_1, \ldots \beta_T$

$\mathbf{x}_T \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$

For $t = T, \ldots, 1$ **do**

a) $\alpha_t \leftarrow 1 - \beta_t, \quad \bar{\alpha}_t \leftarrow \prod_{s=1}^{t} \alpha_s, \quad \sigma_t \leftarrow \sqrt{\beta_t}$

b) $\mathbf{z} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} \leftarrow 0$

c) $\mathbf{x}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\boldsymbol{\epsilon}_\theta\left(t, \mathbf{x}_t, \mathbf{y}\right)\right) + \sigma_t\,\mathbf{z}$

Return $\mathbf{x}_0$

Loss

Mean squared error between noise added at time step t and predicted noise :

$$L(\theta) = \mathbb{E}_{t,\,\boldsymbol{\epsilon},\,\mathbf{x}_t,\,\mathbf{y}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(t, \mathbf{x}_t, \mathbf{y}\right)\right\|^2\right]$$

Similar to guided but with weight w=0:

$$\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, \mathbf{y}) \;=\; (1 + w)\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{y}) - w\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)$$

Model

- MLP with approx 1million parameters
- Initial linear layer (GELU)
- Time step embedding layer
- Series of linear layers (GELU)
- Skip connections

- Input noised data + timestep
- 256-unit hidden layer +learned timestep --> 4 512-unit hidden layers --> 256-unit layer
- 3h training time – once trained 1million events 3 min